4.4. Functional Analysis *(Satisfies EIA 731 FA 1.2 and iCMM PA 4)*

This section details the preferred approach of the Federal Aviation Administration (FAA) to performing Functional Analysis. Functional Analysis is the System Engineering (SE) process that translates stakeholders' needs into a sequenced and traceable functional architecture. This process of analyzing functions provides SE with a functional system description that becomes a framework for developing requirements and physical architectures. Using the Functional Analysis process significantly improves synthesis of design, innovation, requirements development, and integration. The Functional Analysis process provides two key benefits to SE: It discourages single-point solutions, and it describes the behaviors that lead to requirements and physical architectures. Figure 4.4-1 lists the essential elements of Functional Analysis, including the inputs, processes providing input (providers), process tasks, outputs, and processes receiving outputs (customers).

4.4.1. Introduction to Functional Analysis

Systems may be described from at least two different perspectives. One perspective sees the system as a physical architecture with elements that interact with themselves and the system environment in accordance with a predefined process to achieve the system mission. Another view describes the system by the functions that it performs. A system is intended to satisfy predefined functions, with the highest level function defined as the stakeholder need (also the ultimate system requirement or ultimate system function). A *function* is a characteristic action or activity that needs to be performed to achieve a desired system objective (or stakeholder need). A *function name* is stated as an action verb followed by a noun or noun phrase; it is an action that describes the desired system behavior. Examples of common functions include "read book," "eat food," and "go to store." The function occurs within the system environment and is performed by one or more system elements composed of hardware, software, firmware, people, and procedures to achieve system operations. In Functional Analysis, because a function may be accomplished by more than one system element, functions cannot be allocated. Rather, functions are used to develop requirements, which are then allocated to solutions in the form of a physical architecture.

When systems that are being developed radically differ from current ones, the "form follows function" approach is applied. The highest level function, the stakeholder need, is decomposed into lower levels of needed functionality. The functional description is translated into the physical realm by defining requirements from the functions and assigning the requirements to objects within a physical architecture. While, theoretically, function names could be allocated to specific physical architecture entities directly, most times, some combination of two or more architectural entities accomplishes one function.

Process:

Perform Functional Analysis

ID No.: 4.4 (iCMM PA 3,4)
Date: April 13, 2000
Revision Date: August 30, 2006

Next Higher Level Process:

Perform System Engineering

Process Owner:

System Engineering Council

Process Objective:

Provide a framework for developing requirements and physical architectures that significantly improves innovation, synthesis of design, requirements development definition, and product integration.

Inputs

a) FAA Management Decisions
b) Legacy System
c) SEMP
d) Requirements
e) Physical Architecture
f) ICDs
g) Design Analysis Reports
h) Analysis Criteria
i) Constraints
j) Trade Study Reports
k) SE Processes, SE Best-Practices Documentation (SEM), SEBOK

Providers

a) EXT
b) EXT
c) ITP
d) RM, V&V
e) SYN
f) IM
g) SpecEng
h) IA
i) RSK
j) TS
k) MSE

PROCESS TASKS

Beginning Boundary Task

Describe the operational mission, environment, and requirements

- Define top level functions
- Organize functions into logical relationships
- Decompose higher-level functions into lower level functions (iterative)
- Evaluate alternative decompositions
- Document functional analysis baseline

Ending Boundary Task

Deliver completed functional architecture

Lifecycle Phase

- ☑ Mission Analysis
- ☑ Investment Analysis
- ☑ Solution Implementation
- ☑ In-Service Management
- ☑ Disposal

Outputs

a) Functional Architecture
   - Functional flow block diagram
   - N^2 charts
b) Concepts
c) Planning Criteria
d) OSED
e) Constraints
f) Concerns/Issues
g) Tools/Analysis Requirements

Customers

a) RM, SYN, TS, IM, SpecEng , CM, V&V, EXT
b) RM, SpecEng, V&V, EXT, ITP, IM
c) ITP
d) RM, SYN, TS, IM, SpecEng, LCE, V&V
e) TS
f) RSK
g) IA

Figure 4.4-1 Functional Analysis Process-Based Management Chart

4.4.1.1. Functional Analysis Objectives

The Functional Analysis process helps to ensure that:

- All facets of a system's lifecycle, as illustrated in Figure 4.4-1, are covered, from development to production, operation, deployment, and disposal
- All functional elements of the system are described, recognized, and defined
- All system concepts and requirements for specific system functions are related
- Requirements definition is improved
- Product integration is improved
- New and innovative designs and solutions are incorporated

4.4.1.2. Process Overview

The Functional Analysis process examines a system's functions and subfunctions that accomplish the system's operation or mission. It describes *what* the system does, not *how* it does it. Functional Analysis is conducted at the level needed to support later synthesis efforts, with all operational modes and environments included. Each function required to meet the operational needs of a system is identified, defined, and organized into a functional architecture that is used to define system requirements. A functional architecture is a hierarchical arrangement of functions and interfaces that represents the complete system from a performance and behavioral perspective. The process moves to a greater level of detail as the identified functions are further decomposed into subfunctions, and the requirements and physical architecture associated with those functions are decomposed as well. Functional decomposition reduces complexity by allocating functionality and interfaces to more readily understood and managed sublevel functions. This process is repeated until the system is completely decomposed into basic subfunctions, and each subfunction at the lowest level is defined by a valid set of requirements. The interfaces between each of the functions and subfunctions are fully defined, as are the interfaces to the environment and external systems. The functions and subfunctions are arrayed in a functional architecture to show their relationships and internal and external interfaces. **Figure 4.4-2** illustrates the Functional Analysis process flow.

Functions shall be:

- Arranged in their logical sequence
- Clearly defined in their inputs, outputs, and functional interfaces (internal and external)
- Traceable from beginning to end conditions
- Analyzed, determined, and defined for time-critical requirements
- Successively established from the highest to lowest level for each function and interface
- Defined in terms of what needs to be accomplished in verb–noun combinations without describing *how* it is to be accomplished ("implementation free")
- Traceable downward through successive functional decompositions

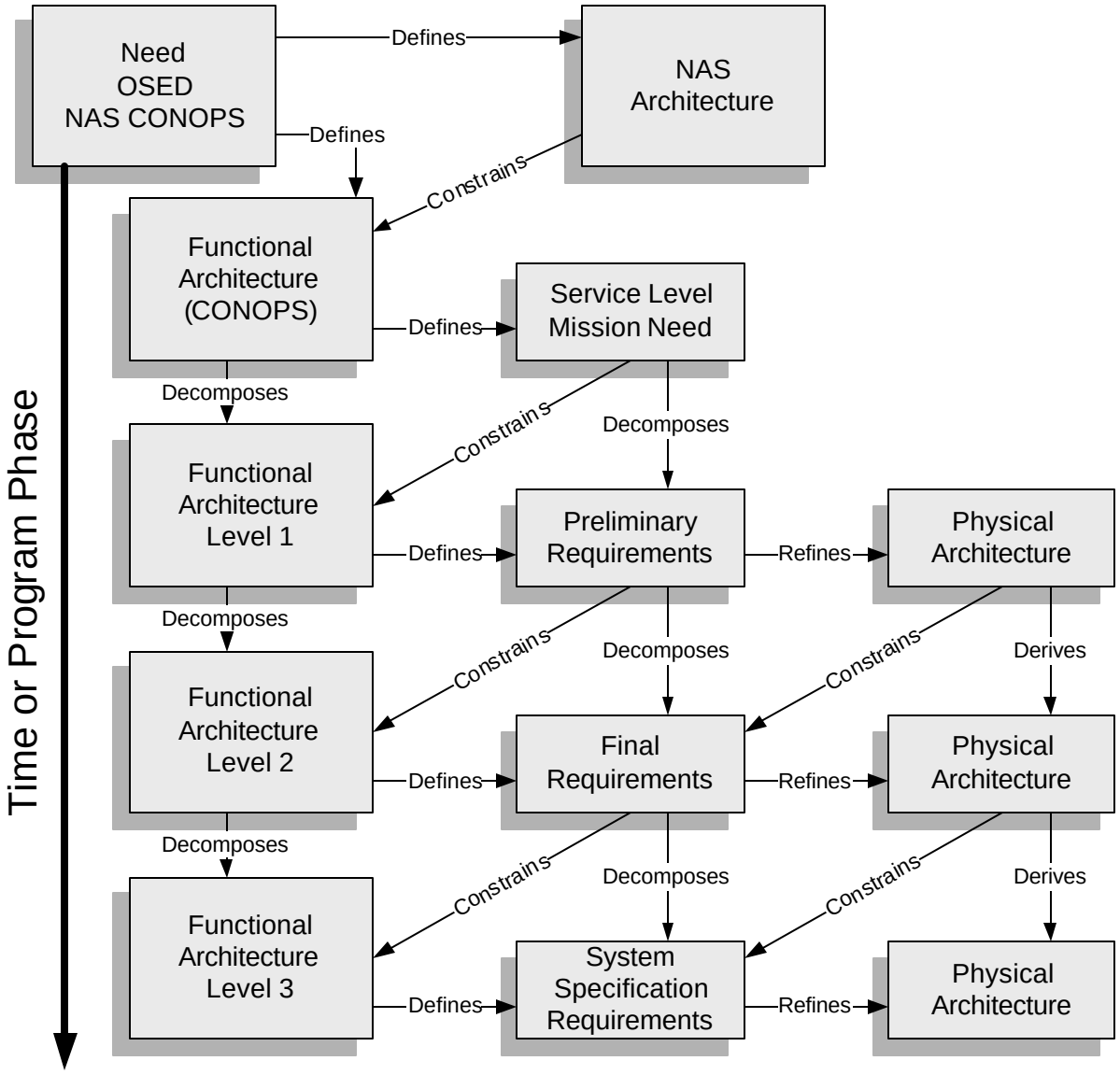

Figure 4.4-2. Requirements Management Process Flow

The Functional Analysis process is conducted in conjunction with Requirements Management (Section 4.3) and Synthesis (Section 4.5) to:

- Define successively lower level functions required to satisfy higher level requirements and to define increasingly detailed sets of the functional architecture
- Define mission- and environment-driven performance requirements and determine that higher level requirements are satisfied
- Flow down performance requirements and design constraints
- Refine the definition of product and process solutions

4.4.1.3. Iterative Process Dependencies

Functional Analysis is an iterative process that works with and depends on the Requirements Management and Synthesis processes. Functional Analysis begins with a high-level

requirement (e.g., a newly identified need) and repeats through successively more detailed layers of decomposition until there is enough insight into the system's desired behavior to completely and correctly define the functional requirements.

Starting with the latest National Airspace System (NAS)- and/or Service-level Concept of Operations (CONOPS), the current NAS-level physical architecture, and the newly identified need, the initial Functional Analysis produces concepts (e.g., a Concept of Use (CONUSE)); see subsection 4.4.5.2 below) for the system (new or modified) that will eventually meet the need. The functions described in a CONUSE, along with identified nonfunctional requirements (e.g., environmental), are used in the Requirements Management process to formally document the new high-level requirement (e.g., in a Service-level mission need statement). At this point in the process, requirements lack sufficient detail to synthesize a physical architecture, so the Synthesis process is not performed.

After completion of the service-level mission need statement during the first pass through the requirements process, the concepts are further decomposed using the Functional Analysis process, as constrained by the requirements. The results of this stage of Functional Analysis are typically captured via one or more diagramming techniques (e.g., functional flow block diagramming (FFBD) and N-squared (N^2) diagramming). This stage of Functional Analysis produces the preliminary draft of the functional architecture and is used to further develop requirements that are documented in the preliminary Program Requirements (pPR). The pPR is used to define the initial draft of the physical architecture during the Synthesis process. The process is repeated until the physical architecture at the lowest system specification level is derived.

At any time during the process, the functions and requirements at a higher level can be reworked as necessary. These changes will then spread downward through the process until the lower levels reflect the changes.

4.4.2. Inputs to Functional Analysis

The stakeholder's needs will be the primary input at the highest level of Functional Analysis for the FAA. This requirement (i.e., a newly established need) is the ultimate function and is used as the catalyst for developing concepts. The initial or highest level concepts for a new or modified system are usually documented in a CONUSE. A CONUSE is primarily a textual document of the results of high-level Functional Analysis efforts. It is usually derived solely from the user's perspective. It is recommended that the CONUSE serve as a baseline for the more detailed Functional Analyses to follow. (Subsection 4.4.5.2 below gives more information on the CONUSE.) Inputs into detailed Functional Analysis will vary depending on the scope of a given effort and the iteration of the process.

Lower level Functional Analysis efforts will have as their input the Service-level mission need, higher level functional and physical architectures, and, eventually, for subsequent iterations of the process, the pPR or fPR. If the output of the Requirements Management (Section 4.3) task is incomplete, the Functional Analysis task reveals missing requirements and helps to refine or clarify other requirements. Additional input includes feedback from stakeholder interviews and functional architecture reviews.

The following is a more comprehensive list of the Functional Analysis inputs:

- FAA management decisions
- Information on legacy systems
- NAS-level (and program, if available) System Engineering Management Plan

- Requirements, such as those contained in the Service-level mission need statement (including defined NAS capability shortfalls and/or needs), requirements documents, specifications, and standards
- Existing physical architectures
- Higher level functional architectures and concepts
- Information on interfaces, including Interface Control Documents
- Design Analysis Reports
- Analysis Criteria
- Constraints

4.4.2.1. FAA Management Decisions

Management decisions that the national, department, or agency level imposes on the system are identified and analyzed for their impacts on the system's intended functionality. Also, program-level management decisions may introduce constraints related to reusing previously developed hardware and software with existing functionality that must be incorporated.

4.4.2.2. Legacy System

Two cases exist in which legacy system information is required as an input to the Functional Analysis process. Case one involves completely replacing an existing system, which means that its existing functionality must be maintained in the follow-on system. Case two involves developing a new higher level system that will incorporate one or more legacy systems (i.e., the legacy system becomes a subsystem within a new higher level system). In either case, lack of any functional documentation for the legacy system may require some reverse engineering to identify the legacy system's functionality and thus derive all the benefits gained from using Functional Analysis.

4.4.2.3. System Engineering Management Plan

This plan lays out the specific system engineering tasks and responsibilities for an organization or program and thus drives Functional Analysis planning efforts.

4.4.2.4. Requirements

The initial high-level requirement, which represents any stakeholder's desire for a new capability or change to an existing system, is often expressed as a "need." Every need requires validation to ensure that it truly addresses a shortfall in capability and/or that it has the possibility to capitalize on a new technological opportunity. This validated high-level requirement initiates the Functional Analysis process and is formally documented in the Service-level mission need as defined in the Acquisition Management System.

Lower level requirements are decomposed from the initial high-level requirement(s) in the pPR, fPR, and specifications and are an input to the Functional Analysis process that constrains or bounds the lower level Functional Analysis efforts.

4.4.2.5. Physical Architecture

A system's physical architecture represents the solution set to defined requirements. A physical architecture is a hierarchical arrangement of hardware and/or software components along with associated interfaces depicting the physical definition of the system. Lower level Functional Analysis work is constrained by a higher level physical architecture. For example, if a radar is

the solution to an aircraft tracking requirement (rather than an optical or thermal tracking device), then the lower level tracking functions will be different than those functions associated with a different solution (i.e., a different physical architecture).

4.4.2.6. Interface Control Documents

Interface Control Documents (ICD) provide the "as-built" solution information to interface with other systems.

4.4.2.7. Design Analysis Reports

Design Analysis Reports (DAR), which document the results of a specific Specialty Engineering analysis, including the rationale, are inputs to the Functional Analysis process. Each DAR contains a description of the system's special characteristics, a list of existing requirements that have undergone the Validation and Verification process (Section 4.12), residual risks, and candidate requirements found as a result of the analysis. The rationale supplementing the DARs includes the scope, ground rules, assumptions, constraints, methods, and tools applicable to the analysis.

4.4.2.8. Analysis Criteria

If the Functional Analysis process requires an analysis or selection of a tool, analysis criteria are captured for that analysis or selection. The analysis criteria for conducting a required analysis are in the Analysis Management Plan.

4.4.2.9. Constraints

Constraints are internal or externally imposed boundary conditions that place limits on the system.

Constraints can stem from various areas, including:

- Management decisions
- Specifications, standards, handbooks, and guidelines
- Policy directives
- Established lifecycle processes
- Physical, financial, and human project resources
- Design limitations

4.4.2.10. Trade Study Reports

In the Functional Analysis process, multiple functional architectures may be produced to accommodate alternatives in accordance with various combinations of constraints. These architectures are then compared using the Trade Studies process (Section 4.6) with the design criteria from Synthesis in order to select the functional architecture that most effectively meets mission objectives. The Trade Study reports provide results of the Trade Studies process comparisons to the Functional Analysis process.

4.4.2.11. SEM Revisions

The System Engineering Manual (SEM) and its revisions are not in and of themselves direct inputs into the Functional Analysis process. However, they do impact the actual conduct of the process. As the process is practiced, feedback from users may necessitate changes to the process. The SEM documents these changes.

4.4.3. FAA's Preferred Diagramming Techniques

The FAA prefers using the complementary FFBD and N^2 diagramming techniques for modeling the functional behavior of a system. A complete functional model must depict both the "control" and "data" aspects of the system simply. The simple FFBD technique captures the control (or the logical) environment of a system, while the N^2 diagramming captures the data environment of a system. Subsections 4.4.3.1 and 4.4.3.2 provide a standardized approach to these preferred techniques and lay the foundation for presenting the actual Functional Analysis process tasks in subsection 4.4.4.

To be sure, there are other diagramming techniques (see subsection 4.4.6.2)—each with its own merits—that can be used (when tailoring has been approved) to capture Functional Analysis results. However, these techniques are more visually complex, or they fail to completely capture enough of the information to completely model a system's functionality.

4.4.3.1. Functional Flow Block Diagrams

The FFBD is a multi-tier, time-sequenced, step-by-step diagram of the system's functional flow. FFBDs usually define the detailed, step-by-step operational and support sequences for systems, but they are also used effectively to define processes in developing and producing systems. The software development processes also use FFBDs extensively. In the system context, the functional flow steps may include combinations of hardware, software, personnel, facilities, and/or procedures. In the FFBD method, the functions are organized and depicted by their logical order of execution. Each function is shown with respect to its logical relationship to the execution and completion of other functions. A node labeled with the function name depicts each function. Arrows from left to right show the order of execution of the functions. Logic symbols represent sequential or parallel execution of functions.

A key concept in modeling functional flow is that for a function to begin, the preceding function or functions within the "control" flow must have finished. For example, an "eat food" function logically would not begin until a "cook food" function was completed. The logical sequence of functions (i.e., the functional flow) describes the "control" environment of the functional model. In addition to a function being enabled, it may also need to be triggered with an input. So, in the example, the "eat food" function is enabled once the "cook food" function is completed, and once it receives the "prepared food" as input. This second aspect—triggering a function—speaks to the "data" environment, which the N^2 diagram captures (see subsection 4.4.3.2 below).

Most system functionality can be modeled using the standard symbols discussed below. If an extended set of symbols is required, then it should be defined in the resulting Functional Analysis Document (FAD) to ensure that all stakeholders are able to accurately interpret the diagrams.

4.4.3.1.1. Function Symbology

A function shall be represented by a rectangle containing the title of the function (an action verb followed by a noun phrase) and its unique decimal delimited number. A horizontal line shall separate this number and the title, as shown in see Figure 4.4-3 above. The figure also depicts how to represent a reference function, which provides context within a specific FFBD. (See Figure 4.4-9 for an example regarding use of a reference function.)

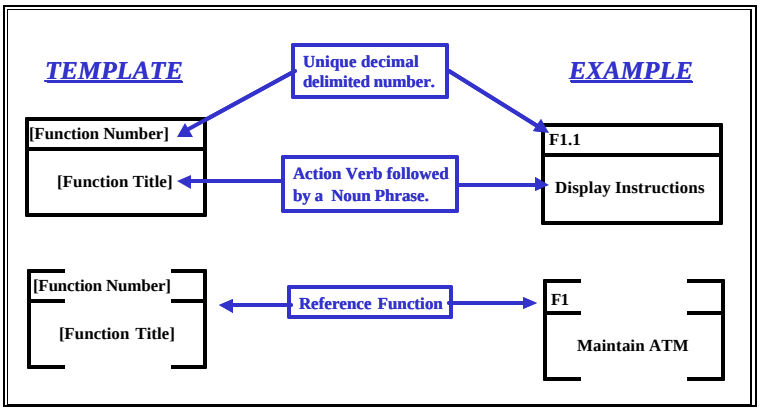

Figure 4.4-3 Function Symbol

4.4.3.1.2. Directed Lines

A line with a single arrowhead shall depict functional flow from left to right (see Figure 4.4-4. Directed Lines

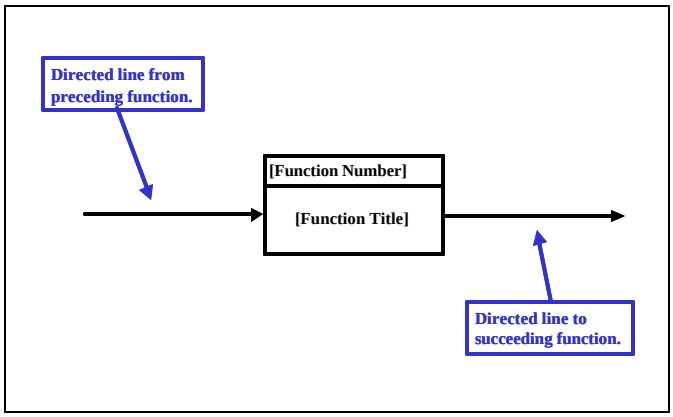

Figure 4.4-4. Directed Lines

4.4.3.1.3. Logic Symbols

The following basic logic symbols shall be used.

AND: A condition in which all preceding or succeeding paths are required. The symbol may contain a single input with multiple outputs or multiple inputs with a single output, but not multiple inputs and outputs combined (Figure 4.4-5). Read the figure as follows: F2 **AND** F3 may begin in parallel after completion of F1. Likewise, F4 may begin after completion of F2 **AND** F3.

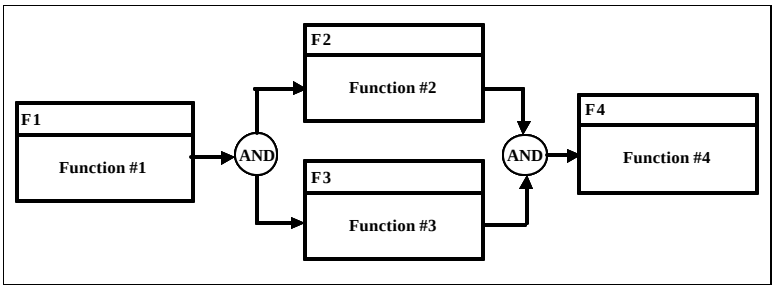

Figure 4.4-5. "AND" Symbol

Exclusive OR: A condition in which one of multiple preceding or succeeding paths is required, but not all. The symbol may contain a single input with multiple outputs or multiple inputs with single output, but not multiple inputs and outputs combined (Figure 4.4-6). Read the figure as follows: F2 **OR** F3 may begin after completion of F1. Likewise, F4 may begin after completion of either F2 **OR** F3.

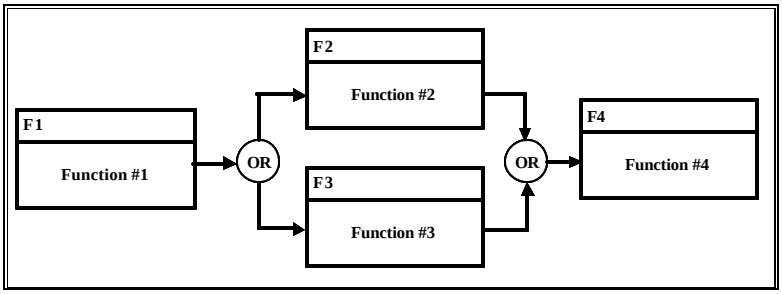

Figure 4.4-6. "Exclusive OR" Symbol

Inclusive OR: A condition in which one, some, or all of the multiple preceding or succeeding paths are required. Figure 4.4-7 depicts **Inclusive OR** logic using a combination of the **AND** symbol (Figure 4.4-5) and the **Exclusive OR** symbol (Figure 4.4-6). Read Figure 4.4-7 as follows: F2 **OR** F3 (exclusively) may begin after completion of F1, **OR** (again exclusive) F2 **AND** F3 may begin after completion of F1. Likewise, F4 may begin after completion of either F2 **OR** F3 (exclusively), **OR** (again exclusive) F4 may begin after completion of both F2 **AND** F3.

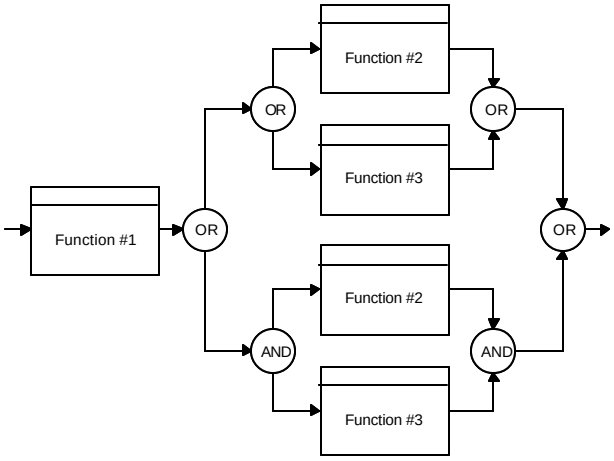

Figure 4.4-7. "Inclusive OR" Logic

4.4.3.1.4. Contextual and Administrative Data

Each FFBD shall contain the following contextual and administrative data:

- Date the diagram was created
- Name of the engineer, organization, or working group that created the diagram
- Unique decimal delimited number of the function being diagrammed
- Unique function name of the function being diagrammed

Figure 4.4-8 and Figure 4.4-9 present the data in an FFBD. Figure 4.4-9 is a decomposition of the function F2 contained in Figure 4.4-8 and illustrates the context between functions at different levels of the model.

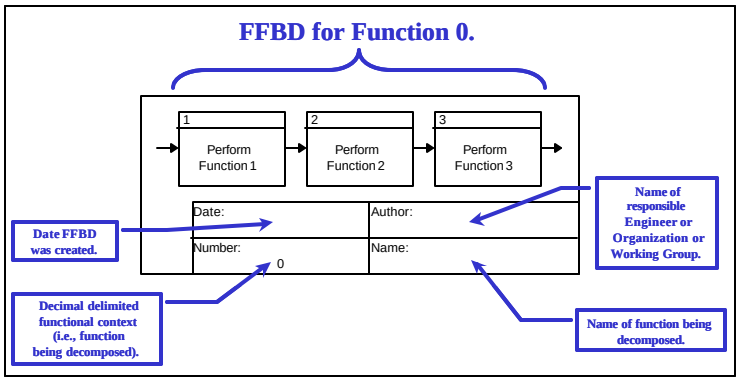

Figure 4.4-8. FFBD Function 0 Illustration

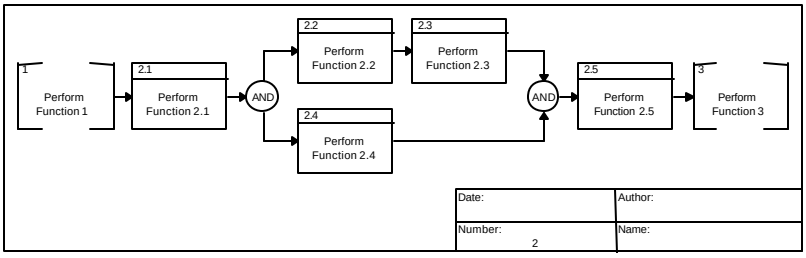

Figure 4.4-9. FFBD Function 2 Illustration

4.4.3.2. N-Squared Diagramming

The N^2 diagram is a visual matrix representing functional or physical interfaces between system elements. It is used to systematically identify, define, tabulate, design, and analyze functional and physical interfaces. It applies to system interfaces and hardware and/or software interfaces. The "N" in an N^2 diagram is the number of entities for which relationships are shown. This N x N matrix requires the user to generate complete definitions of all interfaces in a rigid bidirectional, fixed framework. The user places the functional or physical entities on the diagonal axis and the interface inputs and outputs in the remainder of the diagram squares. A blank square indicates that there is no interface between the respective entities.

Data flows clockwise between entities (i.e., the symbol F1 ↴ F2 in Figure 4.4-10 indicates data flowing from function F1 to function F2; the symbol F2 ↥ F1 indicates the feedback). That which passes across the interface is defined in the appropriate squares. The diagram is complete

when the user has compared each entity to all other entities. The N^2 diagram should be used in each successively lower level of entity decomposition. Figure 4.4-10 illustrates directional flow of interfaces between entities within an N^2 diagram. (In this case, the entities are functions.)

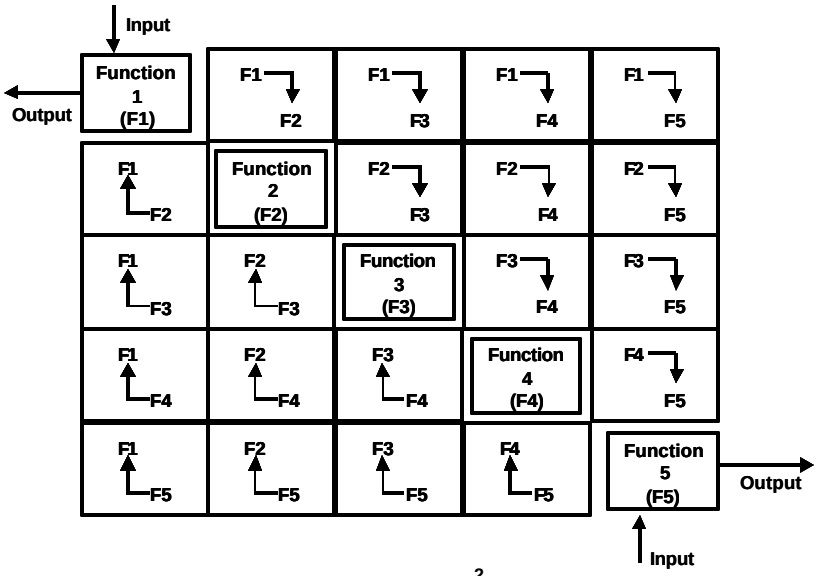

Figure 4.4-10. N^2 Diagram

In the above example, N equals 5. The five functions are on the diagonal. The arrows show the flow of data between functions. So if function 1 sends data to function 2, the data elements would be placed in the box to the right of function 1. If function 1 does not send data to any of the other functions, the rest of the boxes to right of function 1 would be empty. If function 2 sends data to function 3 and function 5, then the data elements would be placed in the first and third boxes to the right of function 2. If any function sends data back to a previous function, then the associated box to the left of the function would have the data elements placed in it. The squares on either side of the diagonal (not just adjacent squares) are filled in with appropriate data to depict the flow between the functions. If there is no interface between two functions, the square that represents the interface between the two functions is left blank. Physical interfaces would be handled in the same manner, with the physical entities on the diagonal rather than the functional entities.

N^2 diagrams are a valuable tool for not only identifying functional or physical interfaces, but also for pinpointing areas in which conflicts may arise with interfaces so that system integration proceeds smoothly and efficiently.

Each N^2 diagram shall contain at a minimum the following contextual and administrative data:

- Date the diagram was created
- Name of the engineer, organization, or working group that created the diagram
- Unique decimal delimited number of the functional or physical entity being diagrammed
- Unique name for the functional or physical entity being diagrammed

Figure 4.4-11 presents the information in an N^2 diagram, which complements the FFBD (Figure 4.4-8 above). Notice that in this illustration, there are no data elements or triggers. Figure 4.4-9 is a decomposition of the function F2 in Figure 4.4-11 and illustrates the context between functions at different levels of the model. Figure 4.4-12 complements the FFBD illustrated in

Figure 4.4-9 and is an example of the diagram's appearance when cells are populated with data.

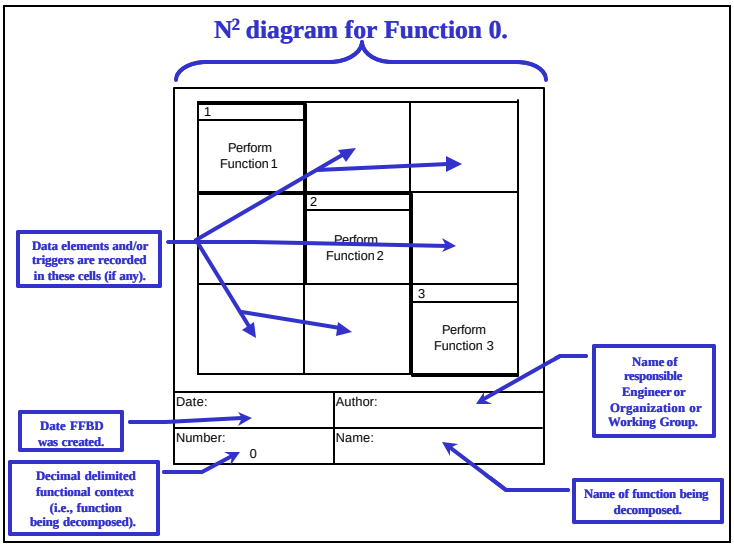

Figure 4.4-11. N^2 Diagram Illustration #1

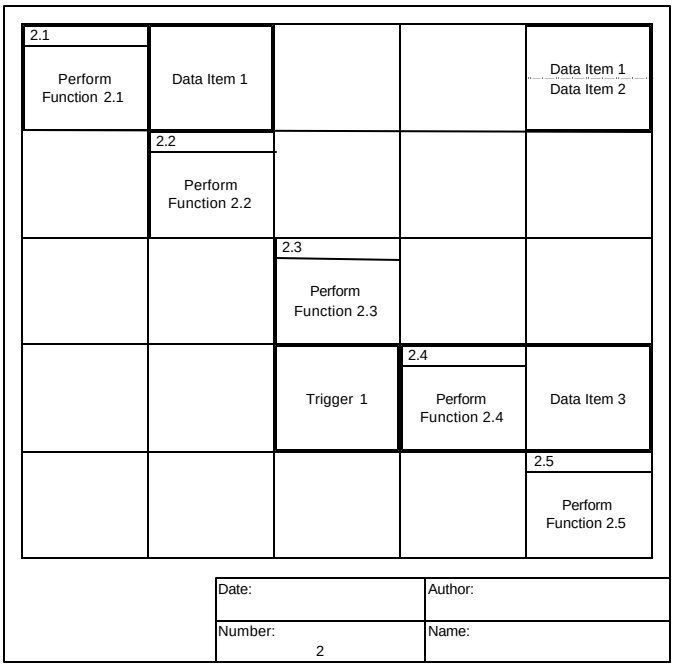

Figure 4.4-12. N^2 Diagram Illustration #2

4.4.4. Functional Analysis Process Tasks

Figure 4.4-1 (Process-Based Management Chart) summarizes the Functional Analysis process, including the five major process tasks. The rest of this section describes these processes within the context of using the FAA's preferred FFBD and N^2 diagramming techniques. These are the same tasks used in developing concepts or an Operational Services and Environmental

Description (OSED), or for alternative diagramming techniques. In generating concepts or an OSED, one simply develops textual descriptions rather than diagrams. (See subsection 4.4.5.2 below for more details.)

To facilitate one's understanding of the Functional Analysis tasks, a functional architecture will be developed from an oversimplified high-level requirement for an autopilot, as follows:

"Avionics shall automatically maintain current altitude, current airspeed, and level attitude upon pilot request."

4.4.4.1. Task 1: Define Top-Level Functions (From Inputs)

Task 1.1 Bound the Problem Space

To define the problem space from a functional standpoint, one must first review all existing inputs to obtain a complete understanding of the top-level missions/functions, environments, requirements, imposed constraints, and boundaries. This understanding of all possible inputs ensures that one will consider the future system's relationship to its environment and external systems during development of the primary functions.

Figure 4.4-13 and Figure 4.4-14 consider the need and create the top-level function called "Perform Autopilot Functions" (outlined in red in the figures). This primary function is named using the guidelines and naming convention described in the "Introduction to Functional Analysis" (subsection 4.4.1) and is the ultimate function that must be fulfilled to successfully accomplish the system's mission. For the purpose of illustration, it is assumed that analysis of other inputs enabled the bounding of the system as captured in the two figures. The boundary is the red outline of function F, "perform autopilot functions." Decomposition of function F will generate all of the functions required within the boundaries of the system to meet the given need. The three other functions—EF.1, EF.2, and EF.3—are external functions.

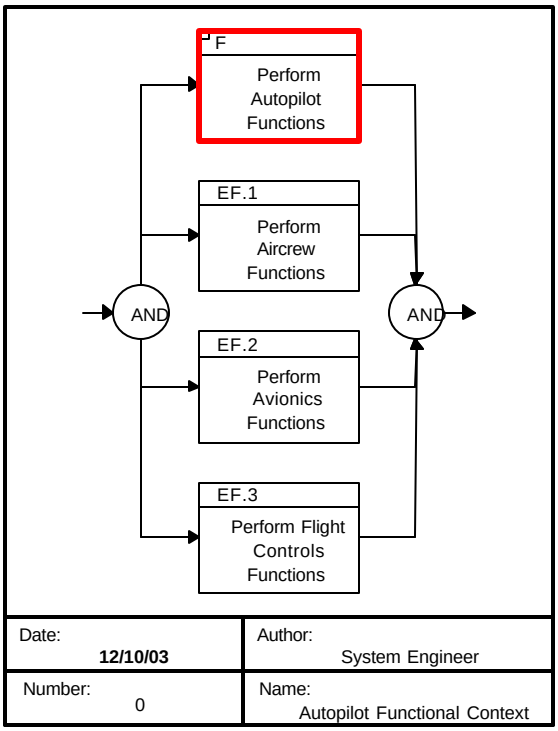

Figure 4.4-13. Autopilot Functional Context FFBD

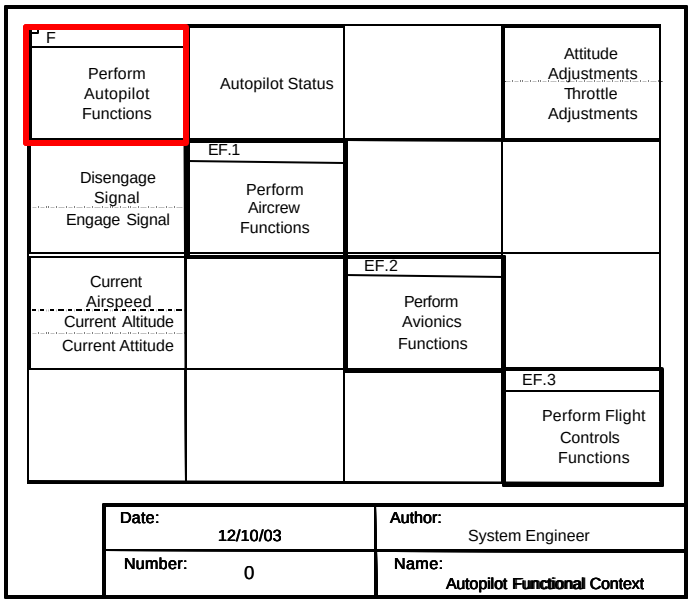

Figure 4.4-14. Autopilot Functional Context N2 Diagram

Note that depending on the iteration of this process, there may exist a higher level FFBD and N^2 that will serve as the functional context diagram. Additionally, due to concurrent engineering efforts, lower level Functional Analysis work may occur in parallel with higher level Functional Analysis work. The lower level working groups are responsible for coordinating their efforts with the higher level working groups.

Task 1.2 Document Assumptions

Where input is lacking, assumptions and issues are documented (see Appendix D) to validate via stakeholders as soon as possible. In reality, if the input was only the need stated for the autopilot example, then essentially all the external functions and data elements in Figure 4.4-13 and Figure 4.4-14 would need to be captured as assumptions and eventually validated.

Task 1.3 Identify Stakeholders

At a minimum the stakeholders shall include:

- The system engineer(s) responsible for the associated service or system
- The system engineer(s) responsible for related cross-cutting disciplines
- The lead for any higher level Functional Analysis efforts

In the autopilot example, stakeholders may include pilot organizations, avionics engineers, and human factors engineers.

Task 1.4 Decompose Top-Level Function

In this task, one must identify and document the highest level functions required to execute the top-level function. The best way to identify these functions is to analyze the system's inputs and outputs captured in the functional context diagrams (see Figure 4.4-13 and Figure 4.4-14). Performing this "thread" analysis, one asks the question, "What must the system do when it receives a specific input?" And, "What must the system do to produce the required output?"

The main criterion for completing this decomposition is development of a comprehensive list of the highest level functions associated with the current iteration of the process that the system

must perform to meet its mission. The list need not be in logical order. Regarding the autopilot example, assume that the following functions are identified:

- Check for aircrew command
- Record baseline altitude
- Record baseline airspeed
- Receive current altitude
- Receive current airspeed
- Record current airspeed as baseline
- Record current altitude as baseline
- Provide autopilot status
- Compare current altitude to baseline altitude
- Compare current airspeed to baseline airspeed
- Make attitude adjustment
- Make throttle adjustment

Task 1.5 Create a Functional Hierarchy

In Task 1.4, there may be functions identified that are lower in level than the actual list of top-level functions to be associated with the current iteration of the process. Figure 4.4-15 is an example of a functional hierarchy using the list of identified autopilot functions.

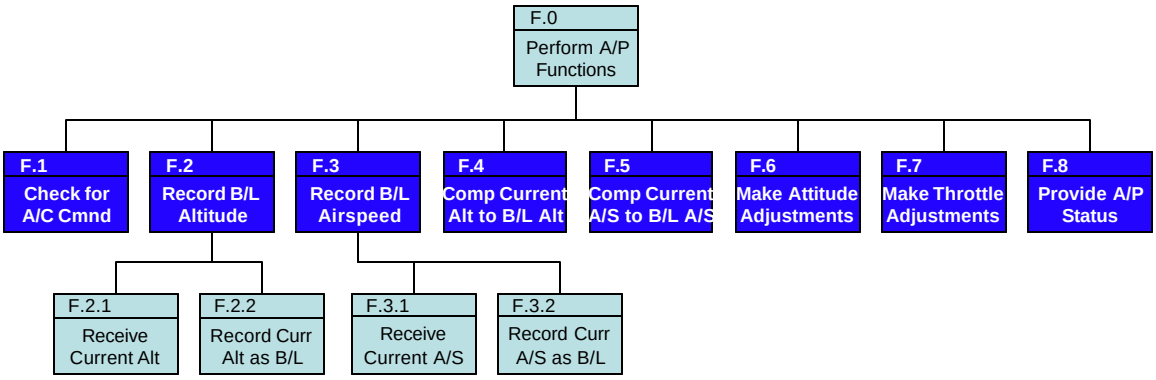

Figure 4.4-15. Autopilot Example Functional Hierarchy

Creating a hierarchy of all the identified functions ensures that the lower level functions are documented for later analysis. In Task 2, only the top-level functions will be considered (e.g., functions 1 through 8 from the list above rather than functions 2.1, 2.2, 3.1, and 3.2). A follow-on iteration of the process will handle the lower level functions

Task 1.6 Create a Lexicon

In creating a lexicon, one defines the functions and data elements identified as providing the required system capabilities. It is recommended that these lexicon entries be defined with an eye toward converting the functional architecture into requirements and requirements into a physical architecture. Developing complementary functional and physical architectures requires multiple iterations between Functional Analysis, Requirements Management (Section 4.3), and Synthesis (Section 4.5).

The lexicon for a given Functional Analysis shall contain at a minimum the following information about a specific term:

- **Name** — either of the following: function name, data element name, or name of the data trigger
- **Type** — a function, data element, or trigger
- **Definition** — a detailed description of the term, including the full scope of its meaning
- **Unique Identifier** — a decimal delimited numeric identifier that facilitates insight into the model's functional hierarchy and data hierarchy (Figure 4.4-15 provides an example of a functional hierarchy, and Figure 4.4-16 provides an example of a data hierarchy.)
- **Source** — the originating source (document, person, organization) that facilitates future validation of any requirements associated with the term

D.1 Autopilot Status
D.1.1 Engaged
D.1.2 Disengaged
D.1.3 BIT Error
D.1.3.1 Altitude Hold Error
D.1.3.2 Attitude Hold Error
D.1.3.3 Velocity Hold Error

Figure 4.4-16. Data Hierarchy Example

The lower level working groups are responsible for coordinating their efforts with higher level working groups to de-conflict on the naming of terms. Thus, lower level Functional Analysis lexicons become subsets of any higher level lexicon.

Affirmative answers to the following questions signify completion of Task 1:

- Has the problem space been clearly identified, including all missions, phases, modes of operation, and interfaces to and from the environment and other systems?
- Are assumptions documented with a plan of action to validate?
- Are all stakeholders identified and listed?
- Has a functional hierarchy been developed to organize the functions identified so far?
- Have all functional elements been properly identified and defined?

4.4.4.2. Task 2: Organize Functions Into Logical Relationships

The function list developed in Task 1 serves as an input to Task 2. This list includes the central functions required for the system to accomplish its mission; but the list functions are not necessarily arranged in a sequence or logical relationship. During Task 2, the highest level functions associated with this iteration of the process (i.e., functions 1 through 8 from the autopilot functional hierarchy) are logically arranged using an FFBD and an N^2 diagram (see **Figure 4.4-17**). The arrangement includes independent functions in parallel and dependent functions in series (e.g., when completion of the upstream function is necessary in order to begin the downstream function). Other diagramming techniques are to be used only when tailoring has been approved (subsection 4.4.6).

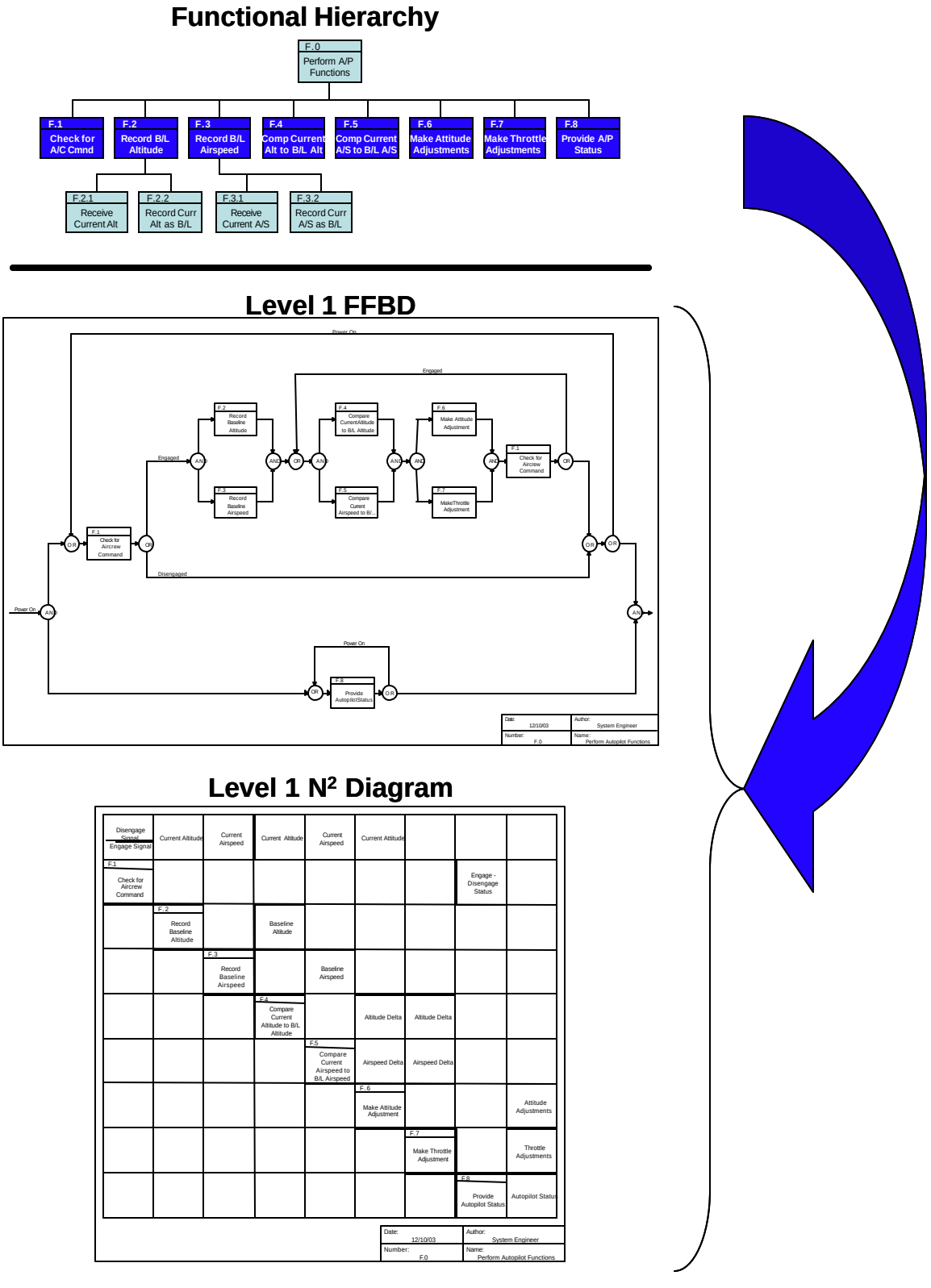

Figure 4.4-17. Logical Arrangement of Highest Level Functions

The following subtasks are detailed and standardized steps to accomplish this second task using the FAA's preferred diagramming techniques.

Task 2.1 Document Assumptions

Document assumptions and issues where input is lacking (see Appendix D) and validate the assumptions with stakeholders as soon as possible.

Task 2.2 Create an FFBD

Create an FFBD (see subsection 4.4.3.1) for the highest level functions currently being worked (iteration dependent) from the functional hierarchy created in Task 1.5. Note that, alternatively, the N^2 diagram could be created first (see Task 2.3).

Tip

In this task, the highest level functions being worked in this iteration of the process are organized into their logical order of flow. Among the questions to ask to determine the logical order of flow are:

- Which functions depend on completion of other functions?
- Which functions depend on data from another function in order to begin execution?
- Which functions could execute in parallel?

Among the rules to remember when creating an FFBD are:

- For a function to begin execution, the preceding function or functions within the "control" flow must have completed execution. For example, in Figure 4.4-18, F.6 and F.7 can only begin execution once F.4 and F.5 have completed execution.
- For a function to begin execution, it may also need to be triggered with the input of data. For example, in Figure 4.4-18, before F.2 can begin execution, it will need "Current Altitude" data as well as F.1 to complete execution. Such data is referred to as a "trigger" and is captured in the N^2 diagram.

Figure 4.4-18 is an example of an FFBD using the level 1 autopilot functions.

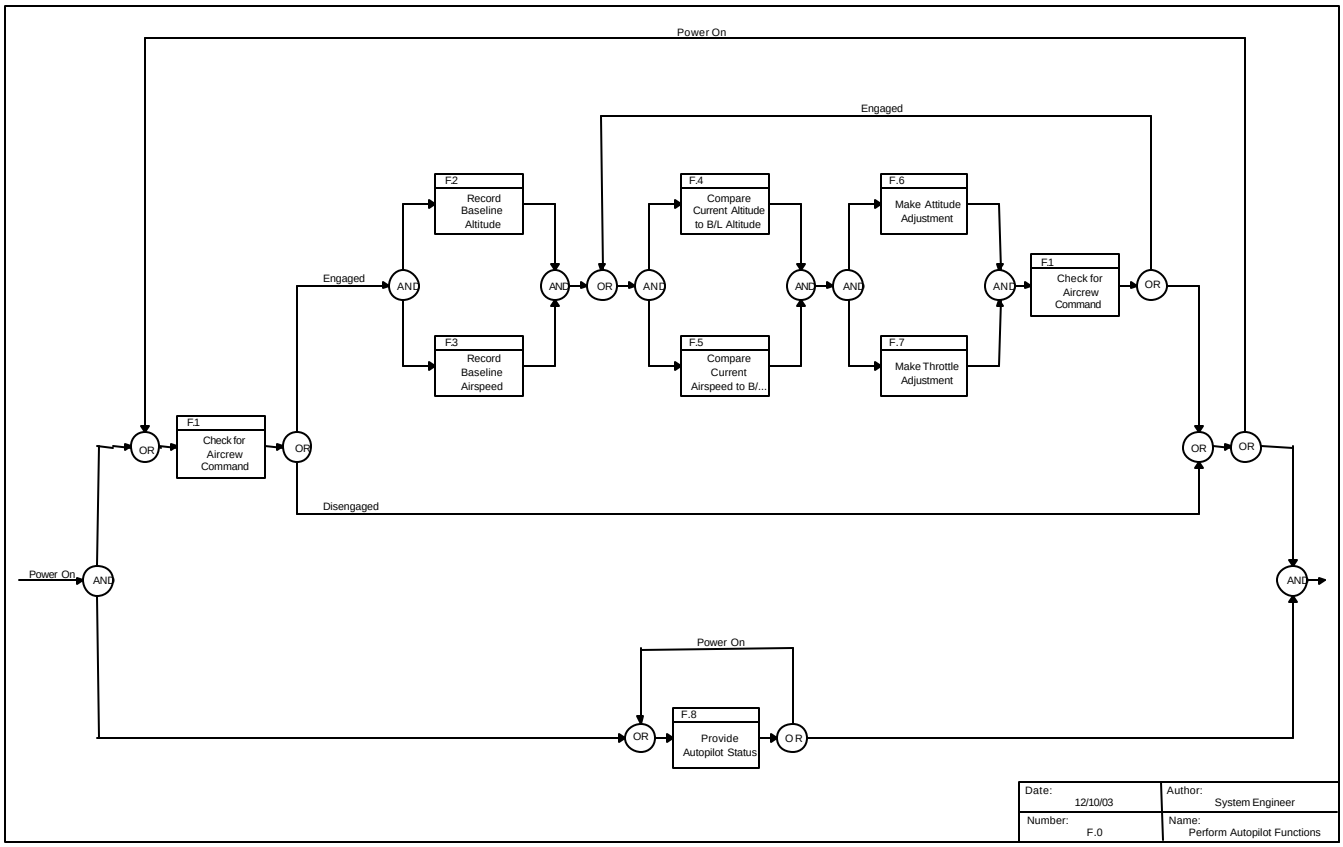

Figure 4.4-18. Autopilot Example FFBD

Task 2.3 Create an N^2 Diagram

Create an N^2 diagram (see 4.4.3.2) using the highest level functions currently being worked (iteration dependent) from the functional hierarchy.

Tip

Among the rules to remember when creating an N^2 diagram are:

- Compare a pair of the functions to determine the data that needs to be exchanged. For example, does F.1 produce any data that F.2 needs? If so, document the data to be exchanged in the appropriate cell. If not, leave the cell blank. Does F.1 produce any data that F.3 needs? And so on.
- Annotate the data items that are "triggers" required for parallel functions to begin execution (use color, or a symbol, or the letter "t"). For example, F.2 in Figure 4.4-19 needs "Current Altitude" data before beginning execution.

Figure 4.4-19 is an example of an N^2 diagram using the level 1 autopilot functions.

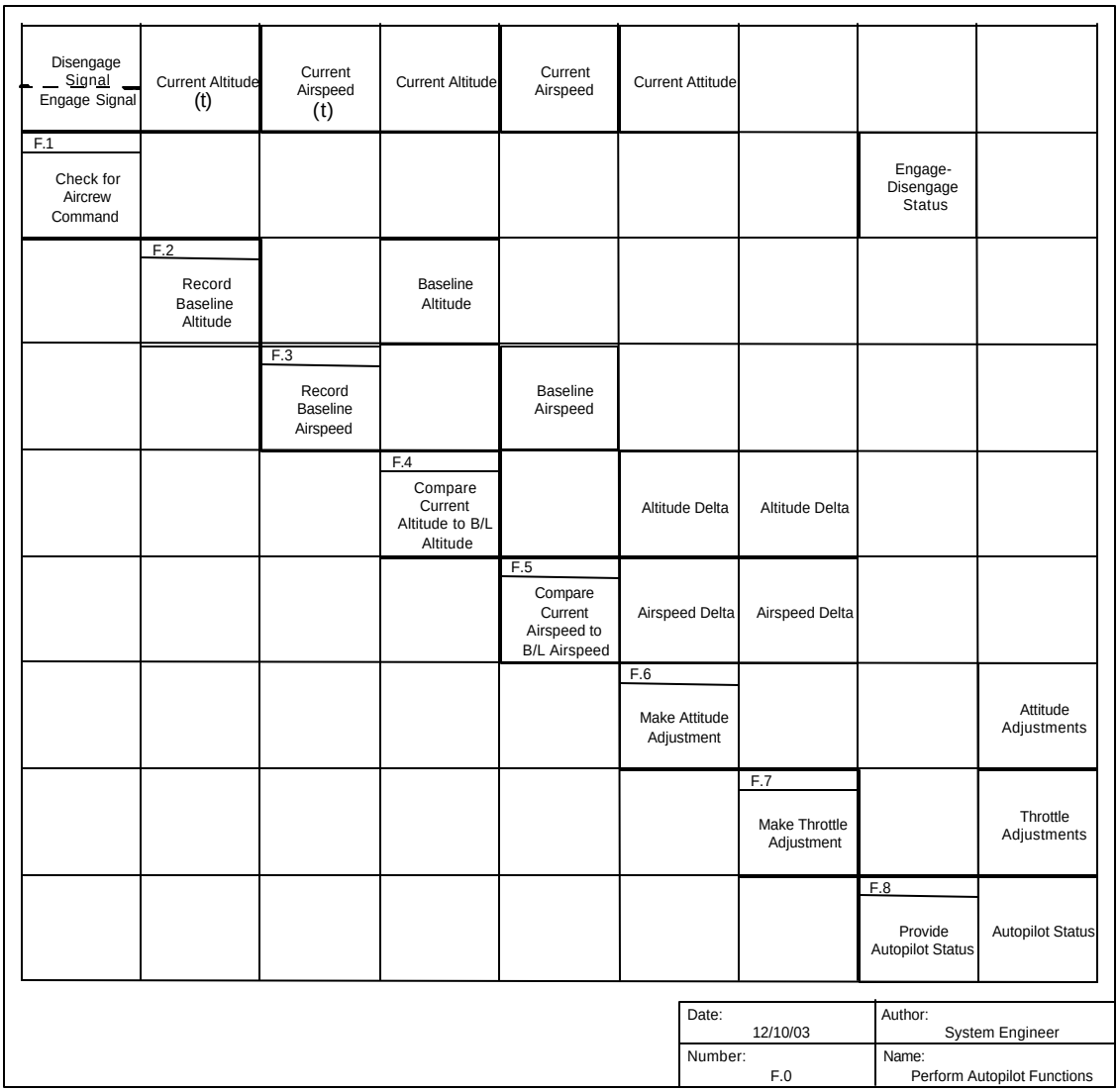

Figure 4.4-19. Autopilot Example N^2 Diagram

Task 2.4 Assign Decimal Delimited Numbers

This task involves assigning a unique decimal delimited number to each function, such as depicted below. Update all diagrams to depict the functions with their assigned decimal delimited number. This numbering system provides context regarding the location of a particular function in the hierarchy.

F.0 Perform autopilot functions (the highest level function)

F.1 Check for aircrew command (level 1 function)

F.2 Record baseline altitude (level 1 function)
- F.2.1 Receive current altitude (level 2 function)
- F.2.2 Record current altitude as baseline (level 2 function)

F.3 Record baseline airspeed (level 1 function)
- F.3.1 Receive current airspeed (level 2 function)
- F.3.2 Record current airspeed as baseline (level 2 function)

F.4 Compare current altitude to baseline altitude (level 1 function)

F.5 Compare current airspeed to baseline Airspeed (level 1 function)

F.6 Make attitude adjustment (level 1 function)

F.7 Make throttle adjustment (level 1 function)

F.8 Provide autopilot status (level 1 function)

Task 2.5 Define Data Flow Items

Define data flow items in the lexicon (see Task 1.6). The lower level working group is responsible for coordinating their efforts with the higher level working group in order to de-conflict on the naming of terms.

Task 2.6 Perform Peer Review With Identified Stakeholders

The newly created FFBD and N^2 diagram, along with the lexicon and any assumptions made, need to be peer reviewed with identified stakeholders. Based on the peer review, the FFBD and N^2 diagram should be modified as necessary.

Task 2 is complete when yes is the answer to the following questions:

- Are all functions in the function list depicted?
- Are all functions written in the verb–noun format?
- Are all functional interfaces depicted graphically?
- Does the depiction show end-to-end functional relationships?
- Are parallel and serial relationships accurately depicted?

At this point, the results of the Functional Analysis effort should start to feed the Requirements Management process (see Section 4.3). The Functional Analysis effort can continue concurrently with requirements analysis tasks. However, since higher level requirements constrain lower level Functional Analysis work, the Functional Analysis effort should not get too far ahead of the requirements effort so as to avoid possible rework.

4.4.4.3. Task 3: Decompose Higher Level Functions Into Lower Level Functions

In this task, higher level functions are decomposed into subfunctions, with specificity increasing at each level of decomposition. Functional decomposition is performed using the techniques described in Tasks 1 and 2 regarding sequence and logical diagramming. The stepwise decomposition of a system basically is a top-down approach to problem-solving. Figure 4.4-21 through Figure 4.4-25 graphically show the execution of decomposition to a level at which the functions have been totally decomposed into basic subfunctions, and each subfunction at the lowest level is defined by its related valid requirement(s). This means that functional decomposition continues as long as there is need to define lower level requirements. When the requirements development process ends, Functional Analysis may cease. Completion of the requirements development process is based on developing the physical architecture. If everything on the lowest tier of all branches of the physical architecture (see Figure 4.4-20) is clearly understood in terms of its needed functionality, then development of the requirements can be completed. Ultimately, good engineering judgment is required to determine whether or not the problem space on the lowest fringe of the physical architecture is fully understood to the extent that a procurement specification or in-house requirements document can be completed from the performance and interface requirements perspective.

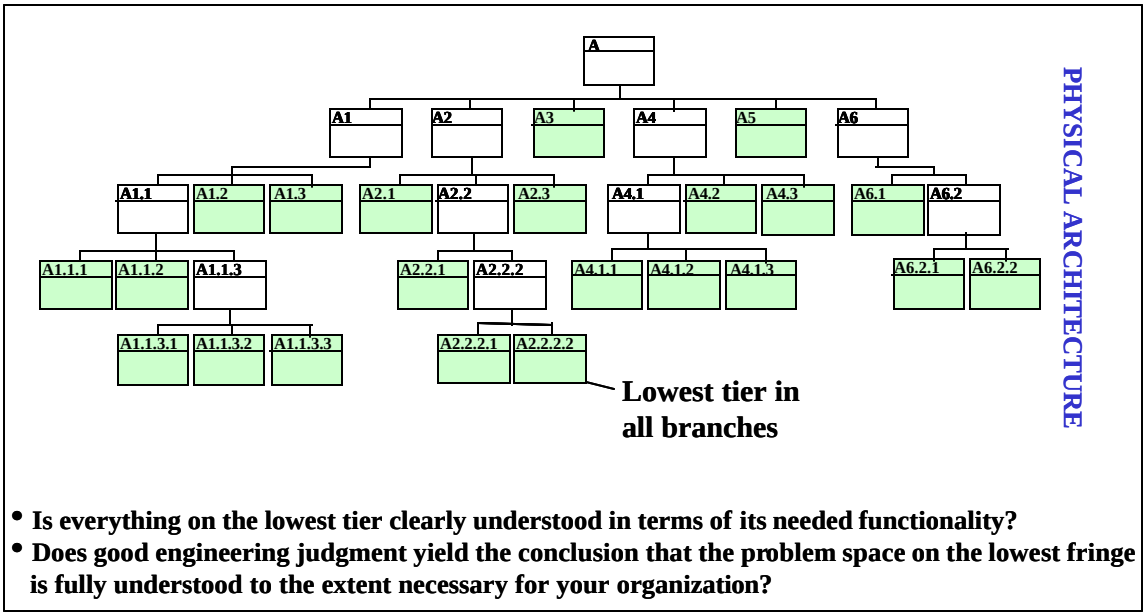

Figure 4.4-20. End of Functional Decomposition

The objective of Task 3 is to develop a hierarchy of Functional Analysis diagrams that describes the functions at all levels of the system. This hierarchy is only a portion of the functional architecture, which is not complete until all requirements and other constraints have been appropriately decomposed.

Task 3 is performed iteratively using the steps and techniques described in Tasks 1 and 2. Since higher level functions exist for this task, the subfunctions are based on the higher level functions developed in the previous tasks. In Figure 4.4-21, function F3 is decomposed into subfunctions labeled as the second level. Next, the functions in the second level are decomposed to the third level. This process continues until all the functions are totally decomposed into basic subfunctions, and each subfunction at the lowest level is completely, simply, and uniquely defined by its requirements. At each level, Functional Analysis feeds Requirements Management (Section 4.3), which feeds Synthesis (Section 4.5), as shown in and further illustrated in Figure 4.4-23 through Figure 4.4-25.

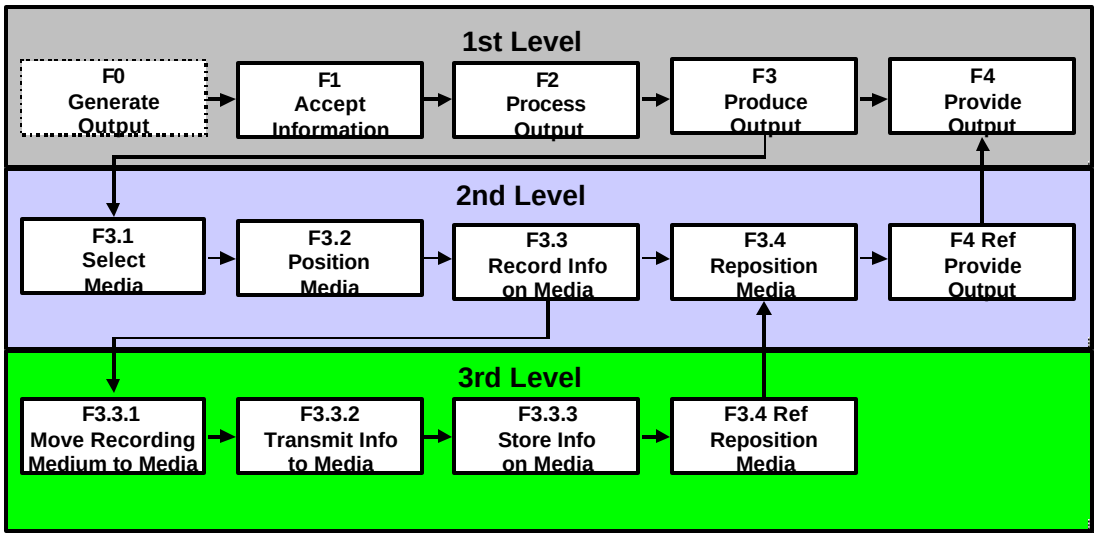

Figure 4.4-21. Breakdown of Higher Level Functions Into Lower Level Subfunctions

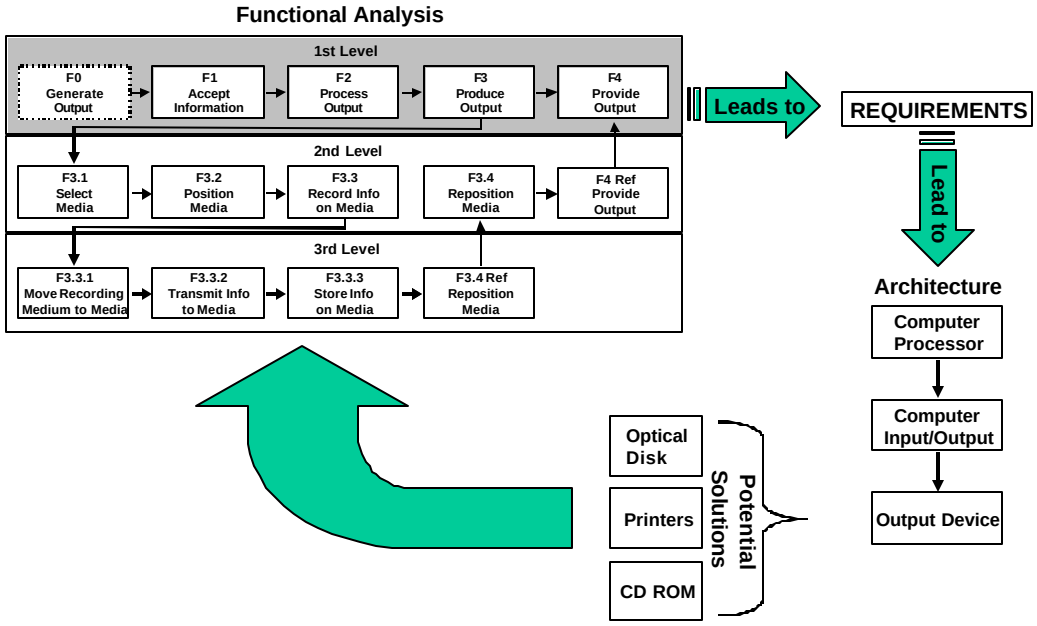

Figure 4.4-22. Functions to Requirements and Requirements to Physical Architectures

Requirements Management and Synthesis detail the process that turns functions into requirements and requirements into a physical architecture. It is important to note that the next Functional Analysis level is bound and framed by the requirements and physical architecture refined from the preceding Requirements Management and Synthesis activities.

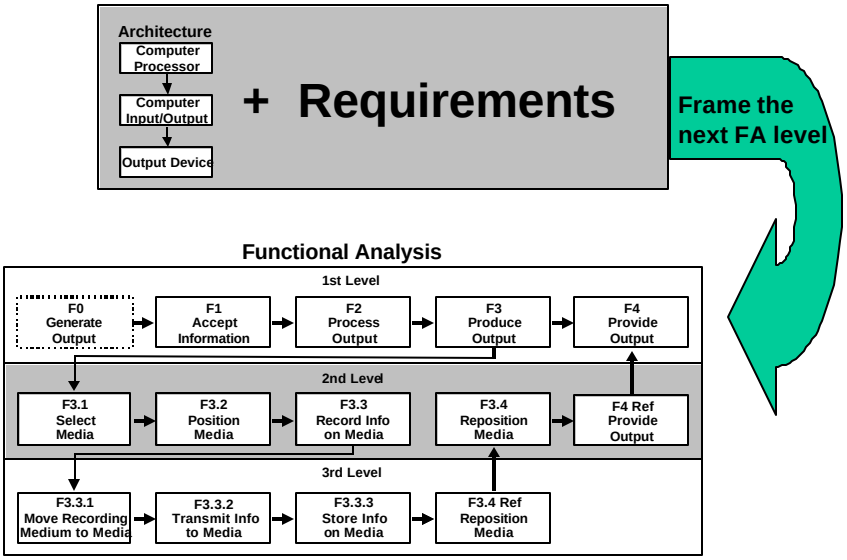

Figure 4.4-23. Requirements and Physical Architecture to the Next Functional Architecture Level

When this process completes one rotation, the Functional Analysis process restarts (see Figure 4.4-23 and Figure 4.4-24) at the next lower level. The process then repeats until each function is totally decomposed into its basic subfunctions, and each subfunction at the lowest level is completely, simply, and uniquely defined by its requirements.

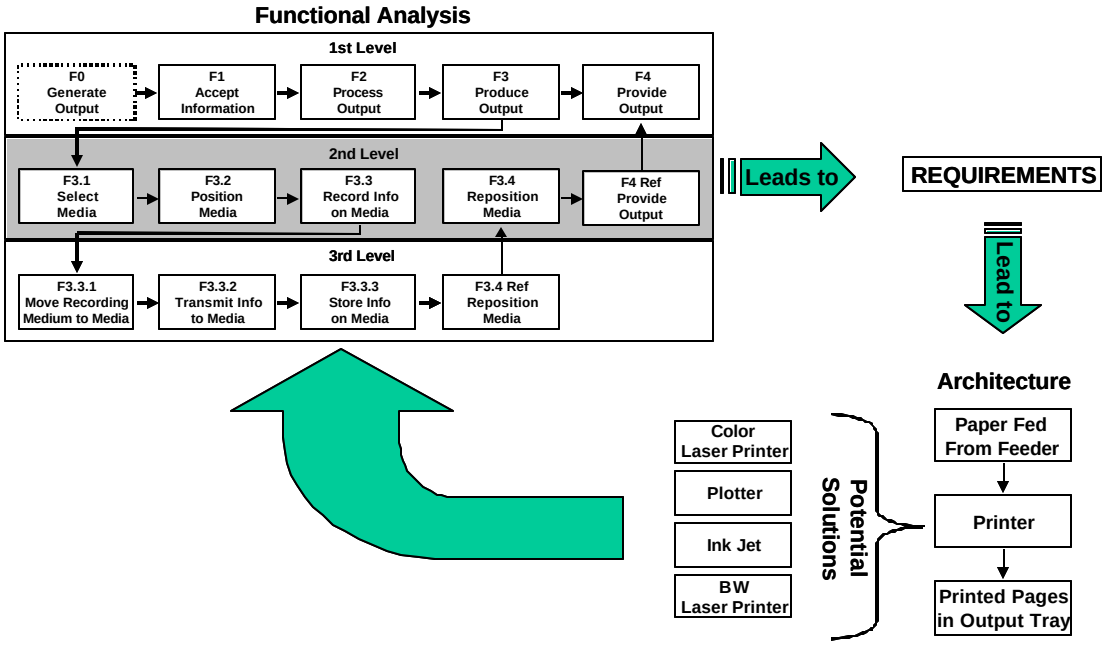

Figure 4.4-24. Repetition of the Functional Analysis Process at Next Lower Level

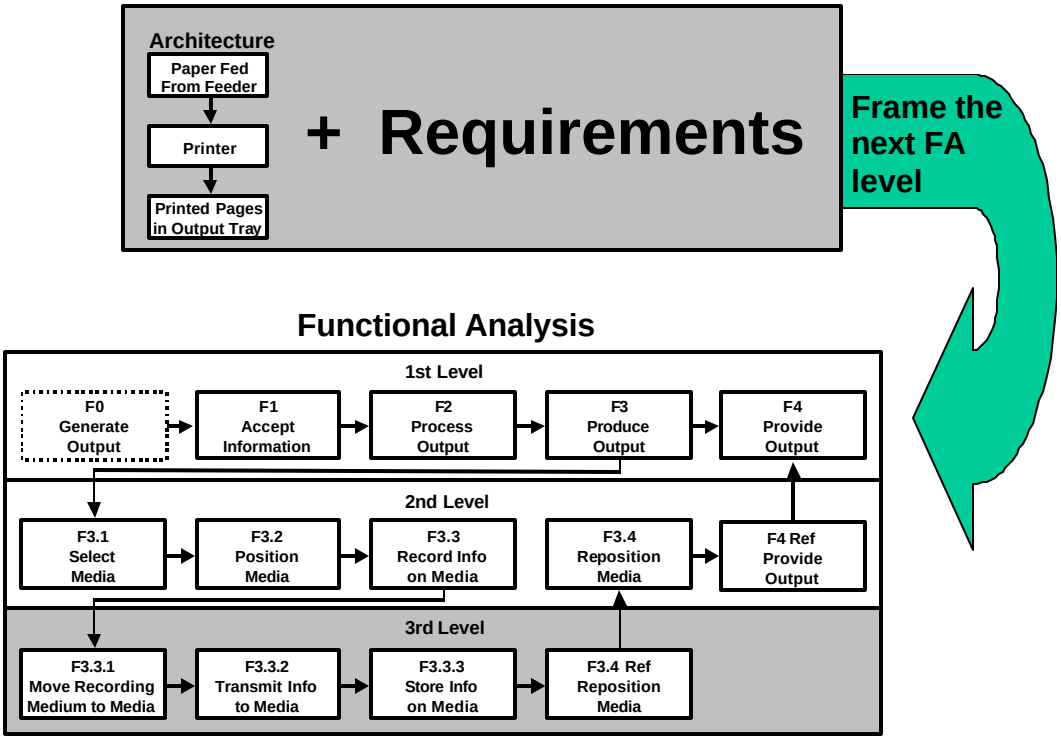

Figure 4.4-25. Preceding Requirements and Physical Architectures

Task 3 is complete when yes is the answer to the following questions:

- Has a complete set of Functional Analysis diagrams been prepared?
- Has each function been decomposed to its lowest level within program needs?
- Is each function completely, simply, and uniquely defined by its requirements?
- Has a description of each function been developed?
- Is the requirements development complete?

4.4.4.4. Task 4: Evaluate Alternative Decompositions

This task evaluates alternative decompositions of functions (functional architectures) and requirements at all levels. These evaluations are necessary because there is no single "correct" decomposition; however, not all decompositions are equal. It is necessary to evaluate alternative decompositions to select the one best suited to the requirements. There are also other reasons to evaluate alternative decompositions. For example, as a result of Synthesis there may surface design constraints such as the desire to use commercial-off-the-shelf (COTS) or non-developmental item (NDI) components. Multiple functional architectures may then be produced to accommodate alternatives in accordance with various combinations of constraints. These are then compared using the Trade Studies process (Section 4.6) with the design criteria from Synthesis in order to select the functional architecture that most effectively meets mission objectives.

The evaluation of alternative decompositions of functions is subjective and depends on personal preference. Task 4 ensures evaluation of other methods to conduct the decomposition. In this task, personal preference and consensus among the stakeholders are factors in selecting the best functional architecture. Any selected functional architecture shall reflect the system's functions; however, variances in the alternative functional architectures may provide a competitive edge to one or more of the alternatives.

By the end of this evaluation process, the requirements for each subfunction at the lowest levels of the functional architecture are allocated via the Synthesis process to hardware, software, interfaces, operations, or a database, and then to a specific configuration item. (See Synthesis, Section 4.5, subsection 4.5.3.4, "Allocate to System Elements—Step 4.") Since it is necessary to verify requirements, the objective of Task 4 is to select those decompositions that promote straightforward requirements that may be validated and verified. (Validation and Verification (Section 4.12) further addresses this issue.) In addition, decompositions that enable a single function to be used at several places within the hierarchy may be identified, which simplifies development.

Task 4 requires "best engineering judgment," as the "goodness" of each functional decomposition is evaluated by measuring the degree that each module displays the following attributes:

- Performs a single function
- Is a logical task
- Leads to a requirement(s) that may be separately validated
- Has a single input point and a single output point
- Is independent within each level of the hierarchy (higher independence enables implementation of the module independent of the other modules)

One should consider using COTS or NDI hardware and software because a subfunction that has already been implemented in a compatible form on another system may be preferred to one that has not.

Task 4 is complete with selection of a final system functional decomposition.

4.4.4.5. Task 5: Document Functional Analysis Baseline

The last task in the Functional Analysis process is documenting the results using a Functional Analysis Document (FAD). Figure 4.4-26 is an outline of the minimum items that the FAD should address. Section 5 and the appendices are the heart of what constitutes the functional architecture. The functional architecture shall be captured in the appropriate enterprise architecture (EA) view(s).

Title Page
- Functional Analysis title
- Document version and date
- Signature block for approving authority

1 Introduction
- 1.1 Purpose
- 1.2 Scope
- 1.3 Rationale
- 1.4 Document Organization

2 References

3 Resources
3.1 Team Members
3.2 Stakeholders
3.3 Software Tools
4 Methodology
5 Analysis
5.1 Assumptions
5.2 Findings
Appendix A — Context Diagrams
Appendix B — Functional Hierarchy Diagram
Appendix C — Functional Flow Block Diagrams
Appendix D — N^2 Diagrams
Appendix E — Lexicon
Appendix F — Acronyms and Abbreviations

Figure 4.4-26. Recommended FAD Outline

Affirmative answers to the following questions signify completion of Task 5:

- Have all of the initial functions been decomposed into subfunctions?
- Do the subfunctions cover the total scope of the parent function?
- Are the functions arranged correctly regarding the dependence of the functions?
- Have all functional interfaces been defined?
- Have any new functional interfaces between initial functions been identified that were discovered during the functional decomposition process? (These may drive new system element interfaces.) If so, have the new interfaces been documented in control sheets?
- Has a Functional Analysis document been prepared to document the functional Baseline?
- Have all functional requirements been identified and decomposed?

4.4.5. Outputs of Functional Analysis

The outputs are static views of the results of the Functional Analysis tasks. As the FAA EA matures, these outputs will be migrated into the various EA views.

4.4.5.1. Functional Architecture

The most common output of the Functional Analysis process is a “living” functional architecture document that contains a tailored combination of the following:

- Functional architecture baseline
- Functional interface list
- Alternative decompositions
- Context diagrams
- FFBDs
- N^2 diagrams
- Other functional descriptions

4.4.5.2. Concepts

In addition to the list above, documents capturing concepts related to the NAS may also be an output of the Functional Analysis process. The two types of concept documents are the CONOPS and the CONUSE. The CONOPS is a description of what is expected from the system, including its various modes of operation and time-critical parameters. The CONUSE is an extension of a higher level CONOPS with an emphasis on a particular NAS system[1] and its operating environment.

There are essentially three levels of concept documents—NAS-Level CONOPS, Service-Level CONOPS, and CONUSE—regarding system engineering the NAS. Figure 4.4-27 shows the documents' hierarchy.

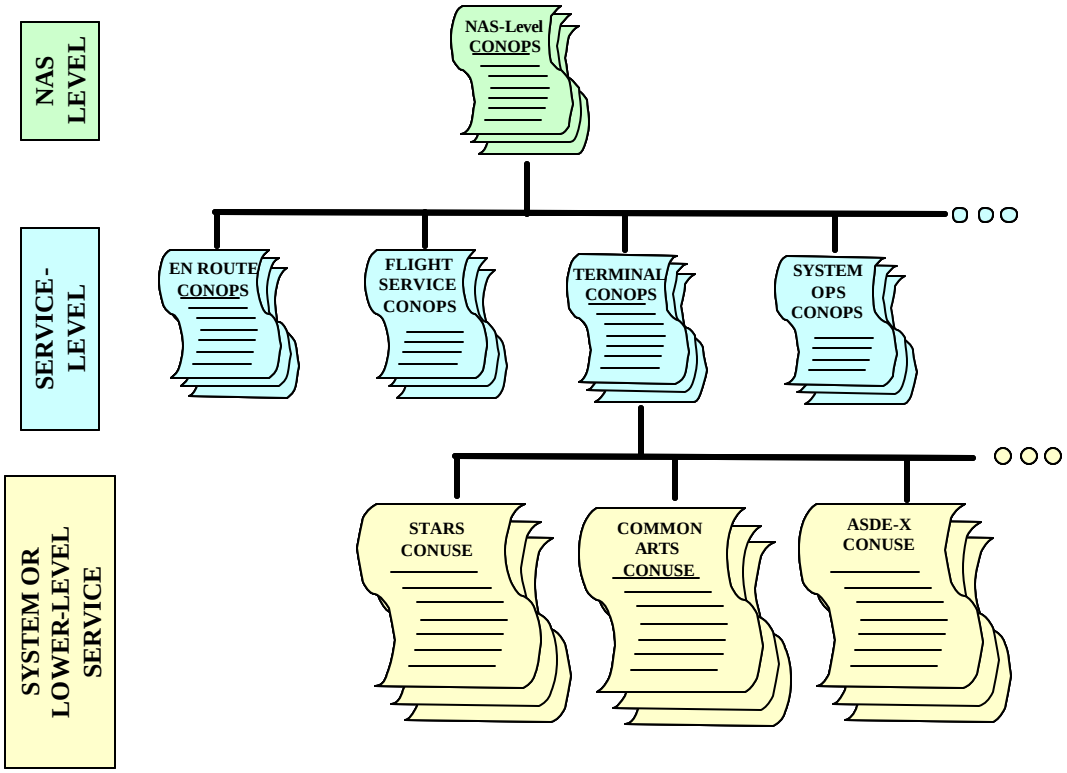

Figure 4.4-27. Concept Document Hierarchy

The NAS-Level CONOPS is a high-level narrative of the user community's desired change with some performance indicators. The document indicates from the user's perspective the desired end-state for respective systems in the NAS. It often uses various operational scenarios to illustrate the desired operational concept. These are characteristics[2] of a NAS-Level CONOPS:

- Describes the integrated operational environment (e.g., communications, navigation, and surveillance/air traffic management)
- Identifies current shortfalls and future needs

[1] Note that in this context and in concert with the SEM's definition of "system," this may be a lower level service rather than a physical system.

[2] These characteristics are adapted from an informal RTCA paper on CONOPS hierarchy.

- Provides a short-, mid-, and long-term perspective
- Identifies the functional requirements
- Identifies the approaches to address current deficiencies and future needs
- Identifies capabilities (without identifying specific technologies)

A Service-Level CONOPS provides conceptual insight into a particular service of the NAS. It gives more detail and in-depth information about the desired operations within the service (e.g., communications, surveillance, etc.). These are characteristics[3] of a Service-Level CONOPS:

- Describes a sub-element of the integrated operational environment
- Elaborates on the capabilities required for the specific service (e.g., communications, navigation, surveillance, etc.)
- Contains all the general categories from the high-level Operational Concept

A CONUSE is an extension of the NAS-Level CONOPS and a particular Service-Level CONOPS, with an emphasis on a particular NAS system[4] and its operating environment. It is more detailed and substantial, but it is still expresses the user's needs regarding a specific system within the NAS. The CONUSE describes functional characteristics for a proposed system from the user's viewpoint; thus, it is essentially a system-level Functional Analysis narrative. It explains the existing system, current environment, users, interaction among users and the system, and organizational impacts. The CONUSE aims to communicate overall quantitative and qualitative system characteristics to the user, buyer, developer, and other organizational elements. The CONUSE aids in capturing requirements and communicating need to the developing organization. Posing the need in the user's language helps to ensure that the user can more accurately express the problem. Subsequently, the system engineers have a better foundation upon which to begin the lower level Functional Analyses, requirements definition, and initial system design. These are characteristics[5] of a CONUSE:

- Written in the user's language in the user's preferred format
- Written as a narrative (in contrast to a technical requirements specification)
- Tells a story using visual forms (diagrams, illustrations, and graphs) and storyboards whenever possible
- Links the user's needs and the developer's technical requirements documents
- Describes the user's general system goals, mission, function, and components
- Evokes the user's views and expectations
- Provides an outlet for user preferences
- Provides a place to document vague and immeasurable requirements (i.e., the user is able to state his/her desire for a fast response or reliable operation); these desires are quantified during the process of developing the requirements specifications and during the flow down of requirements to the physical architecture

---

[3] Ibid.

[4] Note that in this context and in concert with the SEM's definition of "system," this may be a lower level service vice a physical system.

[5] These characteristics are adapted from an informal RTCA paper on CONOPS hierarchy.

Figure 4.4-28 depicts the relationship between the three levels of concept documents. Following are the essential elements of all concept documents:

- Description of the current system or situation
- Insight into the user's environment
- Description of the functions to be performed
- Description of the needs that motivate development of a new system or modification of an existing system
- Insight into the new requirements
- Opportunity for the developer to recommend alternative solutions
- Description of the operational features of the proposed system
- User's view of the requirements

As Figure 4.4-27 and Figure 4.4-28 show, there should be traceability through a common content from a lower level CONUSE to the high-level NAS CONOPS.

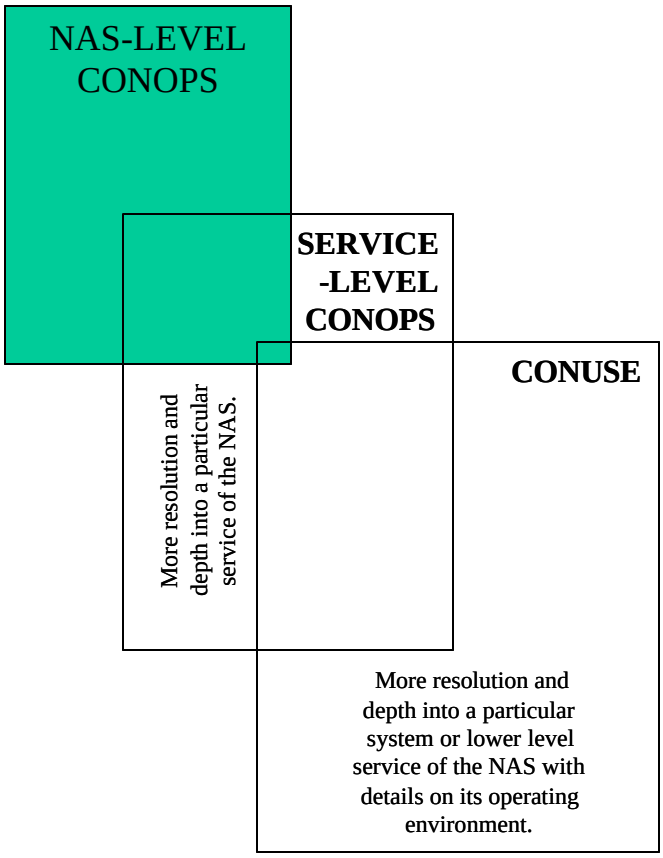

Figure 4.4-28. Concept Document Relationship

Figure 4.4-29 is a recommended outline for concept document content.

CONCEPT DOCUMENT OUTLINE

1. Introduction
   - 1.1 Service or System Identification
   - 1.2 Document Overview
   - 1.3 Service or System Overview
   - 1.4 References
2. Operational Need
   - 2.1 Current Service or System
   - 2.2 Current Support Environment
   - 2.3 Operational Problems
   - 2.4 Objectives and Scope
   - 2.5 Capability Shortfalls
   - 2.6 Existing Operations Requiring Change
   - 2.7 Constraints
   - 2.8 Users
3. Service or System Justification
   - 3.1 Potential Benefit of New or Modified Service or System
   - 3.2 Description of Desired Change
   - 3.3 Change Priorities
   - 3.4 Assumptions and Constraints
4. Proposed Service or System
   - 4.1 Objectives and Scope
   - 4.2 Proposed Service or System Description
   - 4.3 Proposed Support Environment
   - 4.4 Modes of Operation
   - 4.5 Users
   - 4.6 Operational Policies and Constraints
5. Operational Scenarios
6. Impacts
   - 6.1 Impact on Current Operations
   - 6.2 Organizational Changes Required

Appendix A. Glossary and Acronyms
Appendix B. OSED (if available)

Figure 4.4-29. Recommended Concept Document Outline

The guide can be tailored to the document being developed and the information available. All three concept documents essentially contain the same information, but in varying degrees of detail. Thus, some elements in the guide may not be applicable due to the higher level nature of the information being published. The NAS-Level CONOPS is obviously broader in scope than a particular system's CONUSE; therefore, the depth of detail is less in a NAS-Level CONOPS. The breadth of a CONUSE is more focused and thus can contain more details.

4.4.5.3. Planning Criteria

Any planning criteria for performing Functional Analysis throughout the remainder of the program's lifecycle shall be provided to the Integrated Technical Planning process (Section 4.2).

4.4.5.4. Operational Services and Environmental Description

The OSED is a comprehensive, holistic description of the services, environment, functions, and mechanizations that form a system's characteristics.

"What is a System?" A system (as defined in Chapter 2, subsection 2.2) is:

> *An integrated set of constituent pieces that are combined in an operational or support environment to accomplish a defined objective. These pieces include people, hardware, software, firmware, information, procedures, facilities, services, and other support facets.*

The 5M Model illustrated in Figure 4.4-30 represents this system view. Useful system descriptions exhibit two essential characteristics: correctness and completeness. Correctness means that the description accurately and unambiguously reflects the system attributes. Completeness means that all system attributes have been included and that the attributes are essential and appropriate to the level of detail called for in the description. System descriptions that include all 5M Model elements have these two characteristics.

The 5M Model states that there are five basic integrated elements in any system: (1) the functions that the system needs to perform; (2) the human operators and maintainers; (3) the equipment used in the system, composed of the hardware and software; (4) the procedures and policies that govern the system's behavior; and (5) the environment in which the system is operated and maintained.

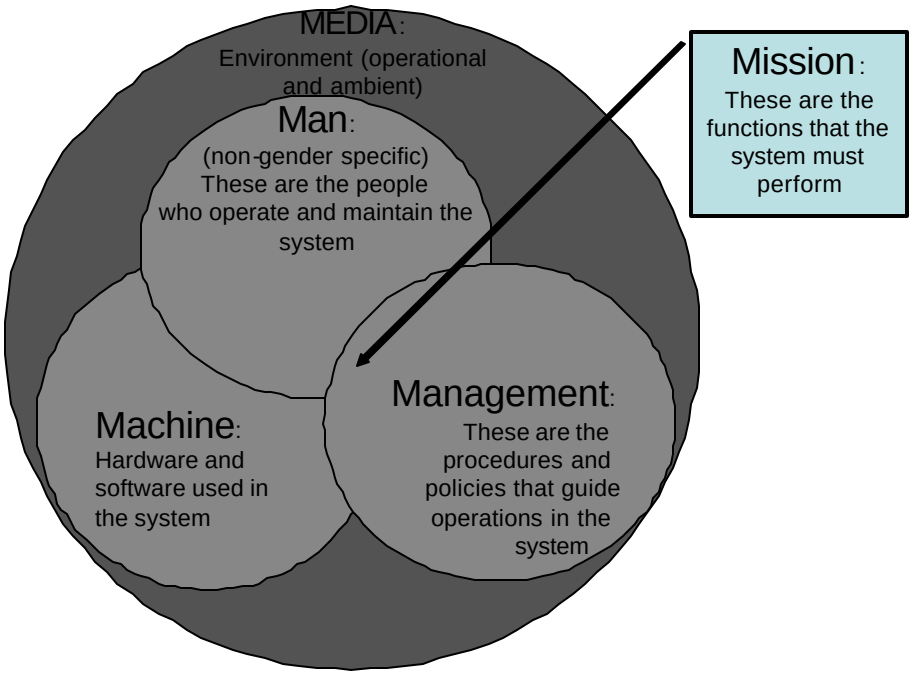

Figure 4.4-30. 5M Model

The document RTCA/DO-264, Annex C, contains detailed guidelines for the OSED for use as a starting point. These guidelines were tailored for the purposes of system engineering in the FAA. An OSED shall have, at minimum, the following information (Figure 4.4-31).

1. Operation Service Description: Summary of the air traffic services and operational context of the new capability from an operator's viewpoint.
2. Functional description or architecture: The functions and functional architecture in accordance with Functional Analysis.
3. Procedures: The existing and new procedures and policies that govern the system's operation or maintenance and includes:
   a. Operational requirements and regulations, including separation minima
   b. Deployment requirements
   c. Operational scenarios
4. Human elements of the system: The operators and maintainers of the system, including information regarding:
   a. Anthropometric requirements
   b. Training requirements
   c. Specific skill-set requirements
   d. Human-system integration requirements
5. Equipment and software: Any known hardware and software that is required for system operation.
6. Environment description: An expression of the various conditions in which the system is operated, including:
   a. Operational: factors such as traffic density and flow, flight phases, traffic complexity, route configuration, type of control, use of visual or instrument flight rules, etc.
   b. Ambient: Refers to visual and instrument meteorological conditions, altitudes, terrain elevations, and physical conditions, such as electromagnetic environment effects, precipitation, icing, etc.
7. Nonfunctional requirements: Any other requirements that are not covered in the other sections and includes, but is not limited to, the following:
   a. Time constraints
   b. Information exchanges
   c. Exception handling

Figure 4.4-31. Guidelines for an Operational Services and Environmental Description

4.4.5.5. Constraints

Constraints on trade studies that surface as a result of performing Functional Analysis are to be provided to the Trade Studies process (Section 4.6).

4.4.5.6. Concerns and Issues

Appendix D contains guidance on concerns and issues as a product of Functional Analysis.

4.4.5.7. Tools/Analysis Requirements

Tools/analysis requirements for performing Functional Analysis throughout the remainder of the program's lifecycle need to be provided to the Integrity of Analyses process (Section 4.9).

4.4.6. Functional Analysis Tools and Techniques

4.4.6.1. Tools

Analysis tools may include but are not limited to general SE and design/simulation aids. Because requirements represent the basic thread through SE, Functional Analysis data shall be interoperable with requirements definition information. The results of the Functional Analysis process shall be captured in order to modify system requirements and other derived products.

Selection of a tool or tools shall ensure that the data is transportable and can be integrated with other related Functional Analysis results. A list of tools that may be used to perform Functional Analysis appears on the International Council on System Engineering Web site (www.incose.org). The FAA's primary functional analysis tool is CORE.

4.4.6.2. Techniques

There are a variety of other diagramming techniques besides FFBDs and N^2 diagrams, and system engineers, for professional development, should become familiar with them. The rationale for this is twofold: (1) There may be rare cases in which the preferred approach does not adequately address FAA needs, and thus the Functional Analysis process must be tailored, with justification and approval, to use an alternative technique to model the system's behavior; and (2) There may be cases in which contractors use different techniques to perform Functional Analysis, and the FAA engineers need to understand what the contractors mean. Among the various other diagramming techniques are:

- Network diagrams
- Time line sequence diagrams
- Hierarchical functional block diagramming
- Integrated Computer-Aided Manufacturing Definition (IDEF) diagrams
- Data/control flow diagrams and context diagrams
- State transition diagrams
- Unified Modeling Language (UML) diagrams

A good overview of various diagramming techniques and their merits appear in a paper, "Relationships between Common Graphical Representations in System Engineering," by Jim Long (available at http://www.vitechcorp.com/infocenter/papers/CommonGraphicalRepresentations_2002.pdf).

4.4.7. Functional Analysis Process Metrics

Candidate metrics used to measure the overall process and products of Functional Analysis include the following:

- Percent of validated assumptions pertaining to the functional architecture
- Percent of identified functions incorporated into the functional architecture
- Percent of functions traceable to validated requirements
- Percent of functional elements clearly and completely defined in a lexicon
- Percent of data elements clearly and completely defined in a lexicon

- Percent of alternatives requiring further de-selection
- Percent of analysis studies completed (schedule/progress)
- Depth of the functional hierarchy as a percentage versus the target depth

4.4.8. References

1. Blanchard, B. *System Engineering Management*. 2nd edition. New York, New York: John Wiley & Sons, Inc., 1997.
2. Blanchard, B., and Fabrycky, W. *Systems Engineering and Analysis.* 2nd edition. Englewood Cliffs, New Jersey: Prentice Hall, 1990.
3. *Certification Considerations for Highly-Integrated or Complex Aircraft Systems*. Aerospace Recommended Practice ARP-4754. Society of Automotive Engineers, 1996.
4. Defense Systems Management College. *Systems Engineering Fundamentals*. Fort Belvoir, VA: Defense Systems Management College Press, 1999.
5. *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*. Aerospace Recommended Practice ARP-4761. Society of Automotive Engineers, 1996.
6. *Guidelines for the Approval of the Provisions and Use of Air Traffic Services Supported by Data Communications*. RTCA DO-264. RTCA, Inc., 2000
7. *Military Standard System Safety Program Requirements.* MIL-STD-88. Department of Defense, 1984.
8. NAS Modernization System Safety Management Program. Washington, DC: Federal Aviation Administration, 2001. (http://fast.faa.gov/toolsets/index2.htm)
9. *NASA System Engineering Handbook*. National Aeronautics and Space Administration, June 1995.
10. Sage, Andrew B., and Rouse, William B., eds. *Handbook of Systems Engineering and Management*. New York, New York: John Wiley & Sons, Inc., 1998.
11. *Systems Engineering Handbook.* Version 2.0. International Council on Systems Engineering, 2000.
12. Yourdon, Edward. *Modern Structured Analysis*. Englewood Cliffs, New Jersey: Prentice Hall, 1988.